【初版】Zoe的Python星际之旅

Zoe

Published
with GitBook

Table of Contents

Zoe的Python星际之旅	0
Week0:旅途开始	1
缘起-为何开始	1.1
入学三问-从这里启航	1.2
初心-是什么打动了我	1.3
开始使用Markdown	1.4
提问的智慧	1.5
Week1：搭建自己的学习环境	2
Sublime配置	2.1
Command Line笔记	2.2
Git	2.3
Git学习笔记	2.3.1
Git安装与工作环境配置	2.3.2
Git操作指南	2.3.3
Gitbook	2.4
Gitbook与Github互推	2.4.1
为Gitbook添加评论	2.4.2
Week2:直面生活	3
工作中两种类型的事情	3.1
工作问题解决思路	3.2
编程思路探讨——小游戏：guess the number	3.3
Running Programs from the Command Line	3.4
用python自动打开网页	3.5
新的教程地址更新	4

[初版]Zoe的Python星际之旅

星际之旅，即将启航。
我们一起，穿越时空隧道，探索好玩的Python吧。

在这里记录，Zoe从零开始的Python旅途。

P.S.

遵照要求，用脚手架重新搭建了教程，
所以以后的教程就在一本新的gitbook里更新啦，

继续 Zoe的Python星际之旅

Week0:旅途开始

不是学编程，而是学习思维

- 学着用一种新的视角去看待世界，新的思维方式去面对世界。

游戏精神，让生命有趣

- 看到好玩的邮件，我觉得我真的喜欢这样的旅途。
- 编程，好玩。其实就是这么简单的开始。

写作，让自己更看清自己

- 勿忘初心。 写着写着，好像就越明白自己想什么了
- Markdown真好用，让思路更清晰。

提问也要认真

- 问之前要自己思考，尝试解决。
- 经过思考后，更能从他人的回答中获得启迪。
- 提问的时候，要有清晰的表达。
- 大妈总是可以反弹出很多问题回来，来回几次后，这种风格也会让我问之前会多想想。习惯其实就是这样培养的。

邮件列表的讨论方式

- 邮件的方式更适合深入的讨论。
- 思想更有体系，而不是碎片化的，而且好的答案可以方便的常常回味，汲取养分。

缘起 - 为何开始

> 你是编程小白，渴望学会编程，退缩、动摇，但坚持三个月后能做啥？——人生苦短，Python当歌

靠近一点点

2015年，还剩下不到100天。想起年初给自己许下的新年誓愿，两个很简单的想法，但是好像今年忙东忙西，离我当时许下的愿望还是那样的遥远。

今年就快要过去了，明年的我会许下同样的愿望吗？
十年后的我呢，是不是还是有同样的愿望，但是却一直没有付诸行动？

我可以再靠近一点点吗？

五年前我没有开始做这件事，那，我能从今天开始吗？
今天，看上去是我余生里最好的时机。

迷茫

我很迷茫。
在我不喜欢的领域工作，我觉得每天都很痛苦。
生活真的是这样吗？我觉得不是，但是我一下子又不知道要怎么蹦出去。
夜晚突然醒来，看着窗外，感觉生命的流逝，会很清晰的感觉到，我不想这样继续下去。

如果，想要改变，那我要去做一些以前不会做的事情，很少做的事情。
我想，我未知的道路，未知的世界，就藏在这些我没做过的事情里，往这个方向走，我会找到新的线索。

我尝试过很多次，也放弃过很多次。
我每次都会欣喜的发现，世界总是给了我很多新的窗口，新的机会，但也会发现自己到后来莫名其妙的又一次次放弃，回到原地。
如果不是因为喜欢，我不会忍不住一次次想要去尝试。
但是，这背后还有一股力量，我并不是很清楚具体是什么，但是却让我又停止。
一股推力和一股阻力在互相纠结，我想再想办法多使一些力气，度过这一关。

另一种尝试

有人和我说过，对待我现在的这份工作，要么就特别勤奋上进，往上爬；要么就自己轻松一点，过简单的小日子。
而我却纠结，我不想上进，也不想轻松。
后来明白，不想往上爬，因为他们的生活其实并不是我喜欢的，只是别人口中听上去很光鲜，但是却无法真正触动我，我会觉得我只是在随波逐流。
我也不想每天不想事混日子，我觉得很浪费，我觉得内心里有一团火，还没有好好燃烧，不想就被磨灭。
我才发现，我想要第三种生活，去投入的探索我更喜欢的事情。

我不想过完这一生，才发现自己只是在扮演别人眼中的乖乖女。
为了在别人眼中的形象，忙着做别人布置的任务，好累。

我已经长大了，我其实已经有能力，也有资源去做一些自己想做的事情了，不是吗？

看到这期报名启动，我又心动了。
年底比较忙，我不在北上广，除此之外，好像也不是那么的难。
没有完美的机会，就让我再任性一下，多探索一下，送我自己一份礼物吧。

渴望

当我拿到船票，开始旅途，我觉得我像又回到了一个欢呼雀跃的小孩子。
踏上探索未知的旅途，我很开心，我觉得自己又活过来啦。

我明白，接下来我还会碰到很多困难，很多纠结，但是我能感觉我的眼睛又重新亮了起来。

之前的我还担心自己会不会学不下去了，每周能抽出这么多时间来吗？
但当真正开始的时候，我发现其实内心里有一种渴望，有时甚至因为时间太少还有点恋恋不舍，这真的有点出乎我的意料。

给我自己一点空间，让自己自由的成长吧。
没有环境，就为自己创造出一个小小环境来，好好呼吸，好好成长。

星际旅途，即将启航。
我渴望这自由的旅途。

入学三问 - 从这里启航

我想知道什么？

> 我现在的基础如何，为什么想学编程？我来参与开智学院的编程课程，希望学到什么，希望课程结束之后，自己可以做成怎样的事情？

为什么想学编程？

- 想学编程，是因为这是我从小其实就想要去做的事情，但是总是时断时续，没有坚持下来。
- 想到年初给自己定下的新年誓愿，不想这一年过去后，觉得还是没有改变，所以想在最后的三个月里加油一把，让这一年结束的时候不会遗憾。
- 我的人生里花了那么多时间去做别人要求的事情，我也想花点时间去学一些自己喜欢的东西，让自己开心一下。

希望学到什么

- 更大的视野。总觉得现在的视野比较狭窄，基本上就是两点一线的生活，有点浑浑噩噩的过日子。但是我觉得，其实在这里互联网+英语的时代，世界其实给了我很多可能，我想去打开这扇窗，看到更大的世界，做点有趣的更酷的事情。
- 解决问题的思维方式。面对问题常常迷茫，纠结，然后就卡住了。希望自己能够有更结构化的思维方式，更有条理的去分析和解决自己遇到的问题，而不是在茫茫的信息碎片中迷失了，忘记了，然后某一天重新在来过，一直在这里打转转。能够把自己的问题，自己拥有的素材、信息以一种更清晰的方式呈现出来，能够更好地利用现有的资源去通往自己想去的地方。
- 快乐和投入。混日子感觉很空虚，我觉得安逸很无聊，我也不想在现有的岗位往上爬，我觉得我想去我更想去的地方。我想重新找回认真做一件事情很投入很充实也很快乐的感觉。
- 转行。想离开这个行业，去做点自己更喜欢的事情，人生苦短，希望自己早点成长为有足够实力离开这个行业的人，有足够实力去做自己喜欢的事情的人。

希望课程结束后能做成怎样的事情

- 对问题的思考和加工更深入，对信息的组织更有条理。能够更有逻辑的解决生活中碰到的问题。自己搜集的信息，自己写的日记也更体系化。
- 建立自己的一个小小网站，作为自己的创作空间，音乐，写作，编程，都是很有意思的事情，不是么。

- 尝试做一个小的app，能够在自己泄气的时候，鼓励自己要倾听内心的声音。
- 尝试把一些自动化的工作，都自动化完成。
- 认识一些好玩的人，学着和大家一起做一些有趣的事情。

我期待学到什么？

> 对于我希望学到的东西，如果自学，可以如何学到？希望开智课程如何助益自己学到这些东西？

对于我希望学到的东西，如果自学，可以如何学到？

- Zed Shaw的Learn Python the hard way和那本好玩的自动化的教程。
- Udacity
- 自己写日记，搜索，思考，钻研。
- Google

希望开智课程如何助益自己学到这些东西？

- 受到更多人的不同视野的启发，感受到身边优秀的人的影响，更有学习动力。
- 百思不得其解的时候，有人能够请教，或者指引我一个方向。
- 更系统化的学习资源。有些东西可能是我这个层面的看不到的，或者完全没有想到的。能够有一张探索地图，让我慢慢去发掘。
- 交流和支持。让我觉得我不是一个人在战斗，和大家共同成长。

如何证明我学到了什么？

> 怎样的结果能说明我已经掌握了自己想学的内容？

- 给自己搭建了一个创作平台，把自己创作的东西可以放进去。——blog或github有关的空间
- 写东西更有条理和逻辑。笔记的记录有经过二次整理，而不是复制和粘贴。——从笔记到整理成手册
- 面对问题可以拆分，并且逐个攻破，而不只是记录感想。——以后写日记不是发泄，而是尝试更多不同的思路，而且更有逻辑和条理。
- 每天的重复任务自己可以写一些小程序自动化。——自动化的小程序
- 生活更有冲劲。拿出更多的时间去做些自己喜欢的事情，而不是当别人眼中的乖乖女。也不是每天消遣。——能够感受到内心更充实，眼神更明亮。

初心 - 是什么打动了我

1. 成为自己想成为的人

我觉得很打动我的一点，是说到成为自己想成为的人。
可能之前说到学习，说到验收成果，很容易想成是完成一个布置的任务，通过一个什么样的测验，却没有想过自己想要做什么，想成为一个怎样的人。
不是一直都说要自由的吗？但是当这自由交到手上的时候，又有些迷茫，有些忐忑，有些兴奋。
我很喜欢这种给人带来无限可能的感觉，好像卸下心中的负担，重新感觉到我又自由了，这种感觉很有趣。
学着感受自由探索的感觉，一切其实就在并不遥远的地方，伸手就可以触动。 我喜欢的生活，其实并没有那么遥远。

> 参考链接：什么是“课”

2. 用一个新的维度看世界

原来，很多觉得不方便不顺手的事情，懂一点编程，就可以自己去尝试改变；
原来，看上去好像很cool的编程，真正做起来也没有那么难；
原来，这件事情还有我以前没有想到的思考维度，不是没有可能，而是之前的我没有看到机会;
原来，这件事情并没有那么困难，只是我并没有去系统的理清它的脉络...
我一直觉得在这个时代，互联网+英语，给了我们太多的机会和可能，我能感受到它们的强大力量，但是却不知道如何去使用它们。
我现在做的事情，只是一种新的尝试，尝试着从一个新的维度去看待这个世界，以及这个世界赋予我们的机会和无限可能。
我觉得这条路上会有很多不一样的发现。

> 参考链接：大妈答疑：如何用编程让大脑二次发育？

3.享受编程的快乐

说了太多正儿八经的话，我觉得我都快忘记那种最初的感觉了啦，就是享受编程的快乐。
还记得最开始接触编程的时候，真的觉得这就是个超酷的玩具，而且所需要的一切东西，居然用一台小小电脑就可以满足。它可以创作出很多很炫很酷的东西，做很多好玩的小玩意。
是呀，最初的感觉就是，编程很好玩。

好玩！
玩性大发，一时兴起，试着去创作一些自己觉得好玩有趣的东西来，做出来自己都觉得超酷超棒，这种感觉很好玩。
我觉得最该记得的，就是这样一种快乐的感觉。
尝试的快乐，探索的快乐，创作的快乐，游戏的快乐。

> 参考链接：大妈对编程学习的建议

开始使用Markdown

为什么要使用Markdown

- 让写作和排版的过程分离，专注于写作
- .md是纯txt文件，跨平台使用
- 可以让自己写作更有逻辑，更有条理
- 文章输出好看且方便

> 学习材料：
> 献给写作者的 Markdown 新手指南

提问的智慧

为什么要认真提问

- 可以得到更好的帮助。
- 自己在提问前有思考，看到大家的回答会更有启发。
- 好的问题所引起的讨论，阅读都是一种享受。

如何认真提问

- 描述清楚自己想解决的问题，想达到的目标。
- 写清楚自己是在什么情境下遇到的问题。
- 问之前自己尝试解决，并写上自己做了哪些尝试，结果如何。
- 写问题思路清晰，排版舒适。

> 参考资料：
> 《提问的智慧》
> 提问的智慧—提问前

Week1：搭建自己的学习环境

敢于交流，敢于提问

- 尝试第一次在邮件列表的提问，特别受启发。觉得这帮小伙伴们太棒了。有这么棒的小伙伴在身边，不认真交流真的好浪费！
- 大妈没有读心术，问问问！其实问出去了，他的回答都很给人启发的。
- 思维碰撞更有趣。自己的视角其实是有一定的局限的，所以有些东西自己琢磨容易钻牛角尖。感受一下和他人交流脑洞大开的感觉，其实很high的。

Git和Github

- 对小白来说，是有门槛的。不过是真的好用，值得好好去学习的。
- 对它有所了解后，思路就从原来的，为什么要用git这么复杂的东西，变成了为什么不用git这么好用的东西，哈哈。

如何学习

- 要操作，不能光看视频。不然打瞌睡，还容易忘。
- 输出是更好的输入。自己写过一遍后，尤其是用自己的方式重新组织语言后，确实印象会更深一些。
- 磨刀不误砍柴工。花点时间看些入门的教程，形成更体系的思维，确实比临时不停的google更好用。

Gitbook

- 第一次有写书的感觉很兴奋。
- 原来写一本自己的小小的书，也没有那么难的。
- 写教程和笔记相比，教程会更有体系一些。

我觉得我选择参加这个课程是对的

- 眼睛更有神了，也不会觉得无聊混日子了，觉得每天有东西要思考，要学习，而且是自己感兴趣的，觉得很有动力。
- 现在觉得上班其实不是累，而是看不到自己的成长和进步，总觉得原地打转，很疲惫。

因为上了一天班很累的情况下，学下编程居然又让我恢复的感觉，出乎我的意料呀。

- 编程给了我一个新的解决方式，我觉得无聊的是个人都能做的事情，其实可以让它们自动化，让自己从重复无聊的工作中解放出来，做点更有意思的事情。
- 总觉得每天在学习一些新的东西，写一些新的东西，探索一些新的世界，创造一些新的东西。很充实。而且这也是我更喜欢的方向。
- 大妈很棒，小伙伴们很棒，有太多值得学习的地方。

Sublime

Sublime

Download Sublime

Launch your editor from the command line

1. Find where the subl command is located. subl comes with Sublime, so it should be in the directory where Sublime is located for you.
   For many people, this is /Applications/Sublime\ Text\ 2.app/Contents/SharedSupport/bin.
   To test this, run ls /Applications/Sublime\ Text\ 2.app/ Contents/SharedSupport/bin.
   You should see the subl command listed.
   If you get the error No such file or directory, Sublime is located somewhere else for you and you'll need to find it.

2. If you do not have a file named .bash_profile in your home directory, create it.
   Because the name of this file begins with a period, it will not appear in most file navigators or when you run ls.
   Instead, run ls -a to see if you have the file.

3. Add the line export PATH=$PATH:/Applications/ Sublime\ Text\ 2.app/Contents/SharedSupport/bin to the end of your .bash_profile.
   If subl was in a different directory for you in step 1, use that directory.

4. Close and re-open your terminal.

5. Typing subl in the terminal should now open Sublime.

参考资料

Setting up Sublime@Udacity

Command Line学习笔记

> 更新日志
>
> - 20151016 增加iomooc卡片 命令行工具下载

还记得"社交网络"那部电影吗?你记得facebook的工程师是怎么与电脑进行交互的吗?

他们用的是一个叫作命令行的东西.
通过命令行,你可以用最快的速度把你的指令告诉电脑. 虽然过去二十年里,图形界面已经有显著发展,但命令行依然是程序员首选的与电脑进行互动的方式,你还没有用过吗?现在就试一试吧.

下载

马上下载最好用的命令行工具:

假如你用的是 Windows 系统---
建议安装运行 Cygwin

假如你用的是 Mac OSX 系统---
建议安装运行 iTerm2

安装好之后建议添加到 Dock 那里,这样可以更方便地找得到. 也可以通过 Spotlight 搜索 iTerm2 来调出.

假如你用的是 GNU/Linux 系统---
用系统自带的命令行工具即可 :)

快捷键

- ctrl c 中断
- ctrl d 退出 ends of file , not any more input coming
- ctrl r 搜索命令
- Tab Completion - 自动完成到可以完成的最远的地方 Tab Again

命令

- ls

- curl http://udacity.github.io/ud595-shell/stuff.zip -o things.zip
- brew install cowsay
- date
- host udacity.com
- echo you rock
- expr 2+2
- uname -a
- history
- hostname - Domain Name System (DNS)
- uptime
- unzip
- cat - concatenate (reading short files)
- wc - word count (lines, words, bytes)
- diff- 比较文件的不同

- cowsay All is not butter that comes from the cow.
- cowsay Hello World!
- cowsay -e ^^ Hello World!
- cowsay -f tux Tux is Linuex\'s mascot!

- man - manual (q - quit)
- man cowsay
- ls -l（“d” directory “-” file）

- apropos working directory -find commands relevant to particular keywords

- ping 8.8.8.8
- bc - calculator

- less - display text one page at a time (D, Space, arrow keys - 翻页;U 上翻页;> 最后一页)
- line number-跳转到某一行
- / search ; n 下一个；N上一个)

- nano - text editor

- pwd - print working directory

- cd - change directory
- cd ..
- cd / (回到root目录；abusolute path) relative path（从working directory开始）
- cd ~ (回到home directory)
- cd (回到home directory)

- mv - move (or rename)
- cp - copy
- mv junk trash
  如果有trash目录 move junk to trash目录
  如果没有trash目录 rename junk to trash

- mkdir - create new directories
- rmdir - remove directories
- rm -r junk 删除非空的junk目录

-man glob（globbing）

- ls *s
- ls app.{css,html} (花括号代表寻找其中任一个)
- ls a?c
- ls be[aeiou]r.png (方括号代表其中任何一个字母) beer bear

- diff -u old.html new.html

参考资料

Udacity
4.3 命令行入门@iomooc

Git

Git学习笔记

Lesson 1

How did viewing a diff between two versions of a file help you see the bug that was introduced?

不用费劲的去比较每一行，而是可以轻松的看到更改的地方，立马让自己回忆起来自己的每一次update。
让发现bug的过程变得简单了很多。
以前居然不知道有这个功能，常常不知道两个文件内容是不是一样的，也不知道哪个文件是更新的版本，更新了哪些地方，因为一个个比较太费劲了。
现在可以让这些费劲的工作让计算机自动帮我完成，我只要根据结果进行判断就ok了，繁琐的工作交给计算机自动化，我自己进行决策就ok。这就是自动化的好处。
以前觉得决策难，是因为要获得决策的数据，和进行决策，这其实是两个部分。
就好像之前一边写文章一边还要排版一样。当我把这两个角色分开，世界就变得更清楚了。
回想起来我日常工作中觉得很累很头疼，应该也是因为有很多角色很多功能混在一起的缘故。
我觉得计算机的练习一个是可以让我学会自动化，一个是可以让我有一双眼睛能够把复杂的事情逐个分解成更简单的东西，不是混为一谈，而是条理清晰，这样解决起来就变得更容易了。
我现在觉得写reflect很有好处，我也逐渐感觉到了计算机学习对思维的训练，因为它真的可以让我看到以前没想到的维度，用我以前没用过的方式去解决问题，比如diff，比如git。
我现在真的觉得，学习python其实并不是为了学习python这种语言去编程，而是去练习一种新的看待问题和解决问题的方式。不是混为一谈，而是一点点的去拆分，分解成更简单的。

How could having easy access to the entire history of a file make you a more efficient programmer in the long term?

我可以看到我是怎么一步步改进的，看到自己的成长，看到自己从前的局限，也可以回忆起自己好的想法。
这是一种成长的印记，记录了自己改进的每一步，也容易唤起自己的记忆，知道自己是怎么走过来的。
我可以快速的找到自己更改的地方在哪里，版本更新了什么内容。
不用担心某些东西消失了，某个版本被误删了，找不到最近版本了，漏掉了之前版本里的好的内容。
不用一遍遍的保存版本1234了。更容易修改了。

What do you think are the pros and cons of manually choosing when to create a

commit, like you do in Git, vs having versions automatically saved, like Google docs does?

好处： 每一次的commit都更有意义，都是一个独立的可运行的作品。
而不是一些无意义的半成品。回顾起来更清晰。
每一个版本更新也更有意义。 坏处： 有时候可能写了很多东西，但是不记得保存了。
需要自己记得要进行手工保存。

Why do you think some version control systems, like Git, allow saving multiple

files in one commit, while others, like Google Docs, treat each file separately?

因为git中的文件常常是相互关联的，一个地方改变，可能引起其他文件的变化；或者一个改变，需要几个文件的配合才能实现。这几个文件是一个联系的整体。
而google doc中的文件基本都是独立的

How can you use the commands git log and git diff to view the history of files?

用git log看自己的更新的不同的版本，注意commit的提示
找到对应的id后，使用git diff作为对比

How might using version control make you more confident to make changes that could break something?

知道自己随时都可以方便的恢复到以前的状态，方便的追溯到问题在哪， 知道自己无论做出多么大的改变，有多么大的错误，都不会有不会挽回的后果。
有一个安全网！我就可以更自由大胆的玩耍，试验，测试。 不会再害怕冒险。
可以去尝试非常大的改变。
我觉得这样一个自由探索的空间，无论是programming，还是生活中，都是我所渴望的。

Now that you have your workspace set up, what do you want to try using Git for?

写书，包括可以进行大的改版
写文章，写作业，写心得
写小程序

Lesson 2

What happens when you initialize a repository? Why do you need to do it?

创建了一个.git文件夹，记录版本信息的变化
如果没有init的话，就无法追踪版本，只是一个普通文件夹，无法使用git的功能

How is the staging area different from the working directory and the repository? What value do you think it offers?

可以把所有文件都放在working directory里面，可以去修改，
但是在上传的时候，可以有选择性的上传几个文件，而不是所有文件。
这样可以保证我能更好地做到one commit per logical change 我可以把握修改成熟的文件上传，或者逻辑相似的文件上传，
同时继续在working directory修改。
这能搞保证我每次都可以做一个接近展示的小成品，而不是一堆草稿。
比如写文章，我可以发布我写得成熟的，或者某几个重大改变，但是其他的草稿我也可以方便的在working directory里面写下来。

How can you use the staging area to make sure you have one commit per logical change?

把这次commit需要的文件加入 staging area
不需要的文件移除staging area
同时使用git diff 确保是这次需要的添加
给了一个可以随时修改的缓冲区
这样在推送前可以有更多考虑

What are some situations when branches would be helpful in keeping your history organized? How would branches help?

试验一些新的功能 做一些探索 测试不同的方向 风格
可以让不同的版本独立发展 又可以容易的共用代码 修改代码
master相当于正式发布版保持稳定
branch可以有很多探索

How do the diagrams help you visualize the branch structure?

语言描述会比较复杂，要动用比喻，有很多分岔，不好描述
但是画图 就会感觉非常直观，会理解是怎么回溯的，一个分支为什么到不了另一个分支
所以有些概念 还是借助图
我决定在我的笔记也放些图

What is the result of merging two branches together? Why do we represent it in the diagram the way we do?

合并两个branch的功能 两个branch的commit按时排序
图的方式更直观

What are the pros and cons of Git’s automatic merging vs. always doing merges manually?

自动的优点是方便快捷 对协作 或者同时开发几个方面很有帮助
缺点是可能有冲突或者冗余
手动当然更优美 结构更清晰
但是确实也更费时 尤其是多人协作时

Lesson 3

What happens when you initialize a repository? Why do you need to do it?

创建了一个.git文件夹，记录版本信息的变化

如果没有init的话，就无法追踪版本，只是一个普通文件夹，无法使用git的功能

How is the staging area different from the working directory and the repository? What value do you think it offers?

可以把所有文件都放在working directory里面，可以去修改，
但是在上传的时候，可以有选择性的上传几个文件，而不是所有文件。

这样可以保证我能更好地做到one commit per logical change
我可以把握修改成熟的文件上传，或者逻辑相似的文件上传，
同时继续在working directory修改。
这能搞保证我每次都可以做一个接近展示的小成品，而不是一堆草稿。
比如写文章，我可以发布我写得成熟的，或者某几个重大改变，但是其他的草稿我也可以方便的在working directory里面写下来。

How can you use the staging area to make sure you have one commit per logical change?

把这次commit需要的文件加入 staging area
不需要的文件移除staging area
同时使用git diff 确保是这次需要的添加
给了一个可以随时修改的缓冲区
这样在推送前可以有更多考虑

What are some situations when branches would be helpful in keeping your history organized? How would branches help?

试验一些新的功能 做一些探索 测试不同的方向 风格
可以让不同的版本独立发展 又可以容易的共用代码 修改代码
master相当于正式发布版保持稳定
branch可以有很多探索

How do the diagrams help you visualize the branch structure?

语言描述会比较复杂，要动用比喻，有很多分岔，不好描述
但是画图 就会感觉非常直观，会理解是怎么回溯的，一个分支为什么到不了另一个分支
所以有些概念 还是借助图
我决定在我的笔记也放些图

What is the result of merging two branches together? Why do we represent it in the diagram the way we do?

合并两个branch的功能 两个branch的commit按时排序
图的方式更直观

What are the pros and cons of Git’s automatic merging vs. always doing merges manually?

自动的优点是方便快捷 对协作 或者同时开发几个方面很有帮助
缺点是可能有冲突或者冗余
手动当然更优美 结构更清晰
但是确实也更费时 尤其是多人协作时

Git安装与工作环境配置

安装Git到Mac上

1.下载

Git下载

2.安装

打开其中.pkg后缀的文件，自动完成安装

3.验证安装

1. 打开Mac的Terminal。
   点击屏幕右上角的搜索图标，输入Terminal。
2. 现在可以输入git的各种命令啦。
   比如输入"git --version",
   如果已经安装成功，会显示正在使用的Git的版本信息。

Git工作环境配置

Setting Up Your Workspace on Mac

Git操作笔记

Git Cheat Sheat

Git Cheat Sheat pdf下载

Q & A

什么是Repository

一些互相关联的文件群。

为什么需要Repository

因为有时候一处修改会影响几个地方。
有时候功能的实现需要几个文件共同完成。
所以git的推送是将这些文件群作为一个整体来推送的。

git log是什么

修改日志。
里面可以看到每次修改的详细情形，比如描述，ID，更改的地方等。

练习步骤

复制Repository

1. 在Terminal中，输入"git clone 你想要复制的的Repository的URL",
   它会复制这个Repository的内容在你现有的工作目录下,
2. 如果你不知道你现在的工作目录是哪里，可以输入"pwd"
3. 如果你想更改自己的工作目录，可以使用"cd 你想使用的文件夹路径"
4. 如果想更好地使用Terminal，建议先学习一些command line的基础知识.

查看git log

1. 使用cd命令，进入你想查看的某个git项目的目录中。可以用pwd命令检验是否到达了对

的目录。
2. 输入"git log"。就可以查看到具体的git log啦。
3. 如果想退出git log的浏览，输入q（代表quit，退出）。

使用**git diff**比较两个版本的不同

1. 根据git log，找到自己想要比较哪两个版本。
2. 复制commit后的一长串字母和数字组合的id。
3. 输入“git diff 第一个id 第二个id”进行比较。
4. 如果觉得id太长，也可以只输入id的前面4个字符。
5. 如果想退出，输入q。

回到上一个版本的状态

1. 输入"git checkout 想回到的版本的id"

Gitbook

Gitbook与Github互推

觉得Gitbook是个好东西，顿时有种想写书的感觉啦。
只是之前以为gitbook和github是自动同步的，因为看上去都是git家族的嘛>_<
结果没想到要实现这个还有自己设置，而且居然一下子还木有设置成功，这个问题就一直闲置在那里几天了。
结果没想到今天试一下居然成功了，记录自己的体会。

解决问题的启示

为什么第一次没解决

- 第一次有发现import tool，提示要输入git url， 其实是要输入book的git url，结果我以为是输入github的url，一直不成功。
- 看到英文界面，加上对github完全不熟悉，有种无从下手的感觉。
- help文档其实很容易看到，但是一直被我无视。

为什么今天尝试成功了

- 学习了一点github知识，对这个没有那么怕了。
- 正好看到有help文档，发现其实有提到这个，就跟着边看边做。

启示

- 学会使用帮助文档。
- 对一个知识不熟悉而害怕，可以先补一下入门知识。
- 当时解决不了没关系，过几天学了一些别的，回过头来可能忽然就会解决了。

> 参考资料：Gitbook帮助文档

步骤：

从Gitbook传送到Github

Transferring content to GitHub

- Use the Import Tool from GitHub
- Enter your GitBook git url,

for example: https://git.gitbook.com/MyName/MyBook.git
(this url can be found in your book settings).
打开网站右上角的“MY BOOKS”，点击你想传送的书名。
打开后，选择右下角的“Settings”。
页面往下翻，可以看到有个Git URL，就是这里要用的地址。

- Enter a name for your GitHub repository.
- Enter your GitBook credentials (you can use your API token instead of your password) when prompted.

连接Gitbook与Github账户

GitHub Integration

When your content as been transferred to GitHub, you can now setup the integration so that GitBook can still build your book from GitHub。

1. Connect your account / Permissions

In your account settings, connect your GitHub account with the correct permissions:

- 点击网站右上角自己的头像，选择Account Setting.
- 页面往下翻，看到有个Github设置的地方，设置自己的Github账户信息。
- 注意这里有个三角形的下拉选框，选择Access to public repositories。

2.Link a book and a GitHub repository

Open the GitHub section in your book settings Enter your repository id (such as: YourGitHubUserName/RepoName) Save your settings

- 打开网站右上角的“MY BOOKS”，点击你想传送的书名。
- 打开后，选择右下角的“Settings”。
- 进入Setting后，点击右边菜单栏最下面的Github。
- 你可以看到设置你GitHub Repository的地方。
- 输入你的repository id (如: YourGitHubUserName/RepoName)
- 保存设置。

3.测试

- 在gitbook里对图书修改，看是否同步到了github。
- 在github里对图书修改，看是否成功同步到了gitbook。
- 如果有问题可以对照帮助文档，继续解决。

为Gitbook添加评论

思考 - 想为gitbook添加评论

- 想到评论就想到之前有在自己的blog上使用过disqus的插件。
- 很多网站也使用了disqus。
- 看到别人的gitbook里有使用disqus的。
- 初步决定使用disqus。或者再搜索一下有没有好的comment pluggin。

摸索

- gitbook的右上角的Pluggin里搜索，有点看不懂。
- google搜索，好像大家都是在本地安装的gitbook，在本地配置的。我想在网站上直接配置。
- 不管了，跟着来，能走一步走一步，先看效果。
- 居然就可以了，明明很多步骤还没做的说，原来只做其中的几步也可以成啊。

启示

- 可以先看看别人做到的人时怎么做的，给自己一些启发。
- 先做自己能做到的，即使还有很多步骤没做也没关系，多走一步又会看见新的风景。

步骤

如果希望别人能够在自己发布到GitBook平台的电子书文章下评论，可以使用第三方评论工具Disqus

注册disqus

- 打开:https://disqus.com/
- 点击"Sign Up",输入邮箱,用户名,密码,创建账号
- 新建站点
- 登录后,点击右上角的齿轮,选择"Add Disqus To Site",填写"Site name",这是book.json里面要填的"shortName","Finish registration".

备注：
"shortName"和注册时填写的用户名作用不同: 用户名是账号的名称,而"shortName"是这个新建站点唯一的名称.一个账号可以有多个站点,也就有多个"shortName".

在网页上使用这个"shortName"以后,这个网页下面的所有评论都会发送到disqus上对应"shortName"的站点. 打开网页时,"shortName"告诉disqus需要加载并显示哪个站点的评论.

插件安装

- Gitbook中，在要编辑的书里， 点击右上角的下拉三角形，选择Edit Book Configuration.
- 系统会自动生成book.jason
- 添加下面的内容。

```
{
"plugins": ["disqus"],
“pluginsConfig”: {
"disqus": {
 “shortName":” “NAME-FROM-DISQUS”
  }
}
}
```

- 将NAME-FROM-DISQUS 改为上面提到注册时写的shortname。

参考链接

Week2:直面生活

Zoe:
直到今天看到弓和箭给六个月前的自己写信的时候， 我才想起，其实我也该为你写一封信。

其实这个教程是写个你的， 只是我有时候会忘记了。
有的时候， 我以为我是要写一个给别人看的教程， 有时候我又以为我是要写一些知识的笔记，
有的时候我又觉得自己是在写日记而已。

其实，六个月前的你在乎这些吗？不在乎。
是的，不在乎。

知识忘记了又怎样，google一下，答案总会有。
编程又怎样，如果只是把它看做一门技能，学了又怎样？
写日记又怎样，写过很多的日记，可是还是迷茫，还是难过，还是找不到出口。

这就是为什么你之前会不停的开始，又不停的放弃。
因为，你觉得没有意义。没有意义。

我知道，你的内心里，是隐隐期盼着一个钥匙，打开新世界大门的钥匙。
你希望，在编程的世界里，可以让你学会推开另一个世界的大门。
可以终于学会解决那些困扰过你的问题，那些困扰着你的生活。

所以我知道，光说编程没法打动你，要给你一把钥匙，打开新世界的大门，感受新世界的思维。
所以我知道，编程不能光只是好玩的虚拟世界的小打小闹，你希望它能给你的现实世界带来真实的改变。

亲爱的，我知道你很害怕，害怕生活会就这样一直下去，
害怕在每个半夜醒来的夜里睡不着，
因为你知道，你的内心并不喜欢这样的生活，你无法再欺骗自己。

很高兴有这样的三个月的时光， 给你写信，一起讨论一些你真正关心的事情，
一起真正的去面对这些问题，并学着用新的思维去解决这些问题。
这一次，不再是心灵疗伤或者是什么心理安慰，
而是，真正去解决这些问题。
想一想，这是不是很酷？

这也是这十多天捣鼓编程给我带来的新的改变，
发现问题，折腾，尝试，搜索，请教，
想尽办法的去解决问题。

你觉得开心吗？
你是不是该觉得，Zoe呀，你可终于开始长大啦？

是啊，小Zoe，我们会长大，用我们喜欢的方式。

你不是说过，你好想要有一个可以让自己自由成长的环境吗？
就让我们一起，学着一起创造一个这样的环境，创造一个真正的Zoe会喜欢的环境。
就让我们在现实的世界里，给自己一份自由呼吸和成长的天地。

编程不是你用来逃避世界所培养的爱好，
而是可以成为你面对生活的有力伙伴。

不要害怕，不要害怕。
其实21世纪的你，拥有很多很棒的资源和能力，真的。
我还知道，你的内心里有一团火焰，不想熄灭，虽然它已经快奄奄一息，但是它还是想要表达出来的，不是吗？

我知道，你不想成为一个无聊的职员。
我知道，你看到他们的生活，觉得自己的未来竟然就像其中的某一种，觉得好可怕，真的好可怕。
你想抱头大哭。

Zoe，我想和你说，其实你有选择的，知道吗？
或许在他们的时代，没有这么多选择的机会，
那个时代没有互联网，没有计算机，没有英语，难以看到外面更大的世界，而现在的你可以。

暂且不说别人，他们变不变也不是我们要操心的事。
我就想说说现在的你，
你想要改变，你觉得不改变很痛苦，
你所处的时代也赋予了你很多可能性，
你也拥有一定的能力去改变，
这样就足够了。

看看是不是真的那么难。
小Zoe，我和你打个赌好啦，我觉得只要你去行动，改变没有你想的那么难。
你一定又要说我站着说话不腰疼了。
好啦，这一切当然也不容易，我也猜到你会碰到很多艰难险阻，不过比起你每天这样要死不活愁眉苦脸的可要好多啦。
你不是还绝望的觉得这辈子就这样了？
哼，我才不信呢。

没关系，反正我们今天就要开始直面生活，用编程思维改变生活了。
我们也不小打小闹，培养什么兴趣爱好，搞什么副业什么的，我们就想真的去改变，
这三个月，让我们好好的学着去改变，去行动，去解决问题，去靠近我们更喜欢的生活。

Zoe Baby，加油！
Zoe Baby，长大成自己喜欢的样子。
Let's go!

工作中两种类型的事情

Part.1 - 是个人都能做的事情

工作中大量的事务性的重复无聊工作，让人深恶痛绝啊。

其实这种事情，随便一个中学生就可以做了，
或者，根本都不用人来做，换个机器来做效果都一样。

我们为什么要花几十年的光阴在这种事情上，数十年如一日的做这种无聊的事情？
无聊就算了，还这么容易被取代，换个人来做其实是一样的。
这样的工作经验有积累的意义吗？
我觉得这就是对自己的一种浪费。多做几年下来，人都会变傻。

正是这种无聊的没有意义的重复的繁琐的事务性工作，耗费了人大量的时间和精力，导致我们没法用更多精力去做一些更重要的事情。

想一想这些破事：

- 反复登陆某个网站几十次，等待网站缓慢的打开，退出，再登陆。
- 每天给别人发几条群发无聊短信。
- 在网站上重复的添加几十条类似的记录。
- 用本子抄写各种东西。
- 每周看各种东西够不够用，调拨。
- 手抄各种登记本，盖章，写错了重抄。
- 反复在一个表中寻找某两个数字对应的数值。
- 登陆系统打印表格，计算机加数字，打电话告诉别人。

我觉得，有些东西我可以靠编程自动化去实现，起码更智能点，
有些东西，我可以提建议去优化系统，虽然不见得马上有效，但不合理的东西终会慢慢淘汰，
有些东西，我暂时无法改变，但是我可以让我生活中无聊的事情比重少一点。

少做点是个人都能做的事情。
太容易被替代。
万一有天被辞退了，都不知道去哪里找新的工作，干这种容易的事情的人可不缺。

Part.2 - 相对需要更多积累的事情

当我觉得有些东西是计算机比较难以自动化的，我觉得要费点脑子的事情，我觉得这才是更值得去做的事情。
但是当有一大堆的事情的事情摆在面前的时候，这种事情就会变得难以识别，甚至因为有点难做而被轻易放弃了。

- 写作
- 演讲
- 主持
- 讲课，培训
- 优化系统
- 优化流程
- 系统的学习

让编程成为我工作中的好帮手

- 能自动完成的事情，交给计算机自动完成
- 不能自动完成，起码给我更多的决策建议，比如我今天要干些什么，怎么干，我容易出错的地方在哪里。不用我在不同的笔记里跳跃来跳跃去寻找要干嘛，容易遗漏。
- 记录中工作中犯的错误，看可否从流程中优化，下次减少这种错误。
- 享受有这样一个好伙伴的快乐，让每天上班不烦躁，心情舒服。
- 花更多时间去做些更让自己开心更充实的事情。

工作问题解决思路

- python
  python自动化的书，最符合我的思路，加上又是python的，值得好好读。
- AutoHotKey
  曾用它实现过类似的功能，可以继续学习。
- Excel
  数据的处理，是不是用excel更方便一点，这个也可以考虑

提醒

时间精力有限，注意**Focus**

不要想着什么都要做，
现阶段先看python那本书，学点编程思维再说。
先把python能解决的事情解决再说。

学会使用别人造的轮子

真正解决问题，不用从零开始，
学会用好这个世界上优秀的人聪明的人提供的好工具好代码好课程。
学会用轮子，就会发现很多问题的解决很容易，而且效果很好。

学会请教和求助

喵的，付费参加这个课程，请好好利用资源，不要浪费！

编程思路探讨—— 小游戏：guess the number

游戏描述-猜数字

计算机先想出一个1-20之间的数字，玩家可以猜6次，看能否猜中数字。

思路一

分析input和output

input

- 游戏者的名字
    - name=input()
- 游戏者猜的数字
    - guess=input()

output

- 向游戏者的问候
    - Hello, what is your name?
    - Hi, xxx , let's play a game - guess the number.
- 游戏引导（介绍数字的范围）
    - Well, I am thinking a number between 1 and 20.
    - Take a guess.
- 游戏提示
    - 数字太大
        - Your guess is too high.
    - 数字太小
        - Your guess is too low.
- 游戏结束
    - 正确猜出（给予赞扬）
        - Good job! xxx, You guessed my number in x guesses!
    - 没有猜出（告知答案）

        - Nope. I number I was thinking of was x.

分析progress

- 用random生成一个随机数字。 secretNumber
- 比较secretNumber和guess的大小，if...elif...
- 可以猜x次。for i in range（）

尝试写游戏

思路二

简单版本1

- 计算机想好一个数字
    - secretNumber=random.randint(1,20)
- 我猜一次
    - guess = int(input())
- 告诉我成功还是失败
    - if...print
    - else...print

简单版本2

- 我猜1次 -> 6次
- 告诉我成功还是失败 -> 告诉我是大了还是小了

简单版本3

- 优化游戏的文字部分，使得它更像个游戏

思路比较

- 思路一思考起来更直观，前期思考会更多些，更能从宏观上把握架构，写出来的东西逻辑性和整体性更强
- 思路二更能马上就开始动手写程序，并且可以马上就运行起来，每次思考过程都有一个独立的可运行的版本，然后不断去改进

我的游戏代码参考

```
import random

print('Hello, what is your name?')
name=input()
print('Hi,  ' +str(name)+ ", let's play a game - guess the number.")

print('Well, I am thinking a number between 1 and 20.')
secretNumber=random.randint(1,20)

for i in range(6):
    print('Take a guess.')
    guess=int(input())

    if guess < secretNumber:
        print('Your guess is too low.')
    elif guess > secretNumber:
        print('Your guess is too high.')
    else:
        break # guess is correct

if guess==secretNumber:
    print('Good job! '+str(name)+', you guessed my number in '+str(i+1)+' guesses!')
else:
    print('Nope. I number I was thinking of was '+str(secretNumber))
```

Running Programs from the Command Line

> 参考资料 Appendix B – Running Programs

Shebang Line

The first line of all your Python programs should be a shebang line, which tells your computer that you want Python to execute this program.

- On Windows, the shebang line is #! python3.
- On OS X, the shebang line is #! /usr/bin/env python3.

You will be able to run Python scripts from IDLE without the shebang line, but the line is needed to run them from the command line.

Running Python Programs on Windows

- create a .bat batch file
    - @py.exe C:\path\to\your\pythonScript.py %*
- Replace this path with the absolute path to your own program, and save this file with a .bat file extension.
- The MyPythonScripts folder should be added to the system path dialog.
- run the batch files in it from the Run dialog.
    - pythonScript

Running Python Programs on OS X

- Save your .py file to your home folder. (Or change the PATH)
- Then, change the .py file’s permissions to make it executable by running
    - chmod +x pythonScript.py
- you will be able to run your script whenever you want by opening a Terminal window and entering
    - ./pythonScript.py.

新的教程地址更新

遵照要求，用脚手架重新搭建了教程，
所以以后的教程就在一本新的gitbook里更新啦，

继续 Zoe的Python星际之旅